**Lab02**

보고서

Lab02 문제해결 프로그래밍 설계 1분반

32207522 양상훈

제출일 : 25/04/08

**실행결과**

**초기 화면**

스크린샷, 텍스트, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**Volcano 클릭**

텍스트, 소프트웨어, 웹 페이지, 웹사이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**Volcano – Detail 클릭**

텍스트, 소프트웨어, 웹 페이지, 웹사이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**USGS에서 더 알아보기 클릭**

텍스트, 스크린샷, 웹사이트, 웹 페이지이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**Earthquake 클릭**

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**Earthquake - Detail 클릭**

텍스트, 소프트웨어, 웹 페이지, 웹사이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**USGS에서 더 알아보기 클릭**

텍스트, 소프트웨어, 웹 페이지, 웹사이트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**소스 코드**

**컨트롤러**

package kr.ac.dankook.ace.lab2.controller;

import kr.ac.dankook.ace.lab2.model.\*;

import kr.ac.dankook.ace.lab2.service.DisasterService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.io.IOException;

import java.util.List;

@Controller

@RequestMapping("/")

public class DisasterController {

@Autowired

private DisasterService service;

@GetMapping("/")

public String home(Model model) {

model.addAttribute("type", "none");

return "lab2";

}

@GetMapping("/list/{type}")

public String showList(@PathVariable String type, Model model) throws IOException {

if (type.equals("volcano")) {

List<Volcano> volcanoes = service.fetchVolcanoData();

model.addAttribute("data", volcanoes);

model.addAttribute("type", "volcano");

} else if (type.equals("earthquake")) {

List<Earthquake> earthquakes = service.fetchEarthquakeData();

model.addAttribute("data", earthquakes);

model.addAttribute("type", "earthquake");

}

return "lab2";

}

@GetMapping("/detail")

public String showDetail(@RequestParam String url,

@RequestParam String type,

@RequestParam(required = false) String name,

@RequestParam(required = false) String date,

@RequestParam(required = false) String alertLevel,

@RequestParam(required = false) String colorCode,

@RequestParam(required = false) String place,

@RequestParam(required = false) Double magnitude,

Model model)

{

model.addAttribute("url", url);

model.addAttribute("type", type);

// Volcano 데이터용 (지진일 땐 HTML(Thymeleaf)에서 사용하지 않을거기 때문에 문제 없음)

model.addAttribute("name", name);

model.addAttribute("date", date);

model.addAttribute("alertLevel", alertLevel);

model.addAttribute("colorCode", colorCode);

// Earthquake 데이터용 (화산일 땐 HTML(Thymeleaf)에서 사용하지 않을거기 때문에 문제 없음)

model.addAttribute("place", place);

model.addAttribute("magnitude", magnitude);

return "detailView";

}

}

**모델 – Volcano.java, Earthquake.java**

**Volcano.java**

package kr.ac.dankook.ace.lab2.model;

public class Volcano {

private String date;

private String name;

private String alertLevel;

private String colorCode;

private String url;

public Volcano() {}

public Volcano(String date, String name, String alertLevel, String colorCode, String url) {

this.date = date;

this.name = name;

this.alertLevel = alertLevel;

this.colorCode = colorCode;

this.url = url;

}

public String getDate() { return date; }

public String getName() { return name; }

public String getAlertLevel() { return alertLevel; }

public String getColorCode() { return colorCode; }

public String getUrl() { return url; }

public void setDate(String date) { this.date = date; }

public void setName(String name) { this.name = name; }

public void setAlertLevel(String alertLevel) { this.alertLevel = alertLevel; }

public void setColorCode(String colorCode) { this.colorCode = colorCode; }

public void setUrl(String url) { this.url = url; }

}

**Earthquake.java**

package kr.ac.dankook.ace.lab2.model;

public class Earthquake {

private String place;

private String time;

private double magnitude;

private String url;

public Earthquake() {}

public Earthquake(String place, String time, double magnitude, String url) {

this.place = place;

this.time = time;

this.magnitude = magnitude;

this.url = url;

}

public String getPlace() { return place; }

public String getTime() { return time; }

public double getMagnitude() { return magnitude; }

public String getUrl() { return url; }

public void setPlace(String place) { this.place = place; }

public void setTime(String time) { this.time = time; }

public void setMagnitude(double magnitude) { this.magnitude = magnitude; }

public void setUrl(String url) { this.url = url; }

}

**서비스**

package kr.ac.dankook.ace.lab2.service;

import kr.ac.dankook.ace.lab2.model.\*;

import org.springframework.stereotype.Service;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

@Service

public class DisasterService {

public List<Volcano> fetchVolcanoData() throws IOException {

List<Volcano> list = new ArrayList<>();

String csvPath = "src/main/resources/static/data/volcano.csv";

try (BufferedReader br = new BufferedReader(new FileReader(csvPath))) {

String line;

boolean isFirst = true;

while ((line = br.readLine()) != null) {

if (isFirst) {

isFirst = false; // 헤더 건너뛰기

continue;

}

String[] tokens = line.split(",", -1);

// volcano.csv는 컬럼이 13개임

if (tokens.length == 13) {

String name = tokens[0];

String date = tokens[2];

String alertLevel = tokens[4];

String color = tokens[5];

String url = tokens[11]; // 실제 링크

list.add(new Volcano(date, name, alertLevel, color, url));

}

}

}

return list;

}

public List<Earthquake> fetchEarthquakeData() throws IOException {

List<Earthquake> list = new ArrayList<>();

String csvPath = "src/main/resources/static/data/earthquake.csv";

try (BufferedReader br = new BufferedReader(new FileReader(csvPath))) {

String line;

boolean isFirst = true;

while ((line = br.readLine()) != null) {

if (isFirst) {

isFirst = false; // 헤더는 건너뛰기

continue;

}

String[] tokens = line.split(",", -1); // 공백도 포함

// earthquake.csv 파일 헤더를 보면 컬럼이 22개이다.

// row 중에서 컬럼이 22개인것만 파싱하고 나머지는 넘어감.

if (tokens.length == 22) {

String time = tokens[0];

double magnitude;

try {

magnitude = Double.parseDouble(tokens[4]);

} catch (NumberFormatException e) {

continue; // 잘못된 숫자는 스킵

}

String place = tokens[13].replaceAll("^\"|\"$", ""); // 큰따옴표 제거

String url = "https://earthquake.usgs.gov/earthquakes/eventpage/" + tokens[11]; // id로 링크 생성

list.add(new Earthquake(place, time, magnitude, url));

}

}

}

return list;

}

}

**lab2.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Lab02-32207522-양상훈</title>

<link rel="stylesheet" th:href="@{/style.css}">

<meta charset="UTF-8">

</head>

<body>

<div class="title-container">

<!-- volcano 이미지 -->

<img id="disaster-image" class="main-image" th:if="${type == 'volcano'}" src="/images/volcano.jpg" />

<!-- earthquake 이미지 -->

<img id="disaster-image" class="main-image" th:if="${type == 'earthquake'}" src="/images/earthquake.jpg" />

<!-- 기본 이미지 -->

<img id="disaster-image" class="main-image" th:if="${type != 'volcano' and type != 'earthquake'}" src="/images/natural-disaster.jpg" />

<h1 class="main-title" id="title">

<a th:href="@{/}" class="main-title-link">Natural Disaster Archive</a>

</h1>

</div>

<div class="container">

<nav>

<a th:href="@{/list/volcano}">🌋Volcano</a> |

<a th:href="@{/list/earthquake}">🏚️Earthquake</a>

</nav>

</div>

<div th:if="${type == 'volcano'}">

<h2>🌋Volcano Activity</h2>

<table>

<thead>

<tr>

<th>Name</th><th>AlertLevel</th><th>Date</th><th>ColorCode</th><th>Detail</th>

</tr>

</thead>

<tbody>

<tr th:each="item : ${data}">

<td th:text="${item.name}"></td>

<td th:text="${item.alertLevel}"></td>

<td th:text="${item.date}"></td>

<td th:text="${item.colorCode}"></td>

<td>

<a th:href="@{/detail(

type=${type},

url=${item.url},

name=${item.name},

date=${item.date},

alertLevel=${item.alertLevel},

colorCode=${item.colorCode}

)}">Detail

</a>

</td>

</tr>

</tbody>

</table>

</div>

<div th:if="${type == 'earthquake'}">

<h2>🏚️2020-2025 Magnitude 5+ Earthquakes</h2>

<table>

<thead>

<tr>

<th>Place</th><th>Magnitude</th><th>Time</th><th>Detail</th>

</tr>

</thead>

<tbody>

<tr th:each="item : ${data}">

<td th:text="${item.place}"></td>

<td th:text="${item.magnitude}"></td>

<td th:text="${item.time}"></td>

<td>

<a th:href="@{/detail(

type=${type},

url=${item.url},

place=${item.place},

date=${item.time},

magnitude=${item.magnitude}

)}">Detail</a>

</td>

</tr>

</tbody>

</table>

</div>

</body>

</html>

**detailView.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Detail View</title>

<link rel="stylesheet" th:href="@{/style.css}">

</head>

<body>

<div class="detail-container">

<h1 th:text="'Detail View - ' + ${type}">Detail View</h1>

<div th:if="${type == 'volcano'}">

<p><strong>🌋 Volcano Name:</strong> <span th:text="${name}">Unknown</span></p>

<p><strong>📅 Date:</strong> <span th:text="${date}">Unknown</span></p>

<p><strong>🚨 AlertLevel:</strong> <span th:text="${alertLevel}">Unknown</span></p>

<p><strong>Color Code:</strong> <span th:text="${colorCode}">Unknown</span></p>

<a th:href="${url}" target="\_blank">

🔎 USGS에서 더 알아보기

</a>

</div>

<div th:if="${type == 'earthquake'}">

<p><strong>🌍 Place:</strong> <span th:text="${place}">Unknown</span></p>

<p><strong>🕒 Time:</strong> <span th:text="${date}">Unknown</span></p>

<p><strong>📏 Magnitude:</strong> <span th:text="${magnitude}">Unknown</span></p>

<a th:href="${url}" target="\_blank">

🔎 USGS에서 더 알아보기

</a>

</div>

</div>

</body>

</html>

**style.css**

.container{

display: flex;

flex-direction: column; /\* 위 -> 아래 정렬 \*/

justify-content: center; /\* flex-direction: column 이므로 세로 방향 + 가운데 정렬 \*/

align-items: center; /\* flex-direction: column 이므로 가로 방향 + 가운데 정렬 \*/

}

.nav{

display: flex;

flex-direction: row;

width:50%; /\* 부모 요소, 혹은 화면 전체의 너비의 절반\*/

justify-content: space-between; /\* 가로 방향 + 여백으로 요소들 나열 \*/

}

.main-image{

width: 100%; /\* 부모(container)에 맞춰 꽉 채움 \*/

max-width: 400px; /\* 최대 너비 제한 \*/

height: auto; /\* 가로 크기에 맞춰 비율 자동 조정 \*/

display: block; /\* 이미지가 줄 하나 다 차지하게 만들기 \*/

margin: 0 auto; /\* 이미지가 가운데로 정렬됨. display: block과 세트로 자주 씀 \*/

}

.main-title{

font-size: 2em;

margin-top: 10px;

margin-bottom: 0;

text-align: center;

}

.main-title-link {

text-decoration: none;

color: inherit;

display: inline-block;

}

h3{

cursor: pointer;

}

.focus{

color: green;

}

table{

border-collapse: collapse;

text-align: left;

line-height: 1.5;

border-top: 1px solid #ccc;

border-left: 1px solid #ccc;

margin : 20px 10px;

}

th{

background-color: #f2f2f2; /\* 밝은 회색 배경 \*/

width: 147px;

padding: 10px;

font-weight: bold;

vertical-align: top;

color: #153d73;

border-right: 1px solid #ccc;

border-bottom: 1px solid #ccc;

text-align: center;

}

td{

width: 349px;

padding: 10px;

vertical-align: top;

border-right: 1px solid #ccc;

border-bottom: 1px solid #ccc;

text-align: center;

}

.hidden{

display: none;

}

/\* detailView \*/

.detail-container {

max-width: 600px;

margin: 40px auto;

background-color: #fff;

border-radius: 12px;

box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);

padding: 30px;

text-align: center;

}

.detail-container h1 {

font-size: 2em;

color: #333;

margin-bottom: 25px;

}

.detail-container p {

font-size: 1.1em;

color: #555;

margin: 12px 0;

}

.detail-container strong {

color: #222;

}

.detail-container a {

display: inline-block;

margin-top: 20px;

padding: 10px 18px;

background-color: #4CAF50;

color: white;

border-radius: 8px;

text-decoration: none;

transition: background-color 0.3s;

}

.detail-container a:hover {

background-color: #388e3c;

}

**코드 설명**

**csv파일 만들기**

volcano.csv 파일 만들기

<https://volcanoes.usgs.gov/hans-public/api/volcano/default> 에서 JSON 파일을 받아옴

<https://convertcsv.com/>에서 JSON을 CSV파일로 자동 변환

earthquake.csv 파일 만들기

2020 – 2025 까지의 진도 5 이상의 지진 csv 파일을 만듦

1980 – 2025 까지 만들 계획이었지만 결과가 7만건 이상이고 2만건 이상인 경우에 csv 파일이 생성이 안되었기 때문에 위와 같이 만들었음.

https://earthquake.usgs.gov/earthquakes/search/ 여기에서 csv 파일을 만들 수 있음.

**html 및 css 작성**

lab2.html은 타임리프 + lab01에서 사용하였던 css 스타일 이용하였음.

detailView.html은 타임리프 + 카드 형식으로 스타일 만들어 보았음.(css에 추가)

detailView에서 화산, 지진 모두 USGS에서 더 알아보기를 클릭하면 USGS링크로 새 창이 만들어지고 상세 설명을 확인할 수 있도록 함.

**코드의 전체 흐름**

**예시)**

localhost:8080에서 volcano를 클릭하는 경우 -> localhost:8080/list/volcano로 이동

DispatcherServlet이 DisasterController에 매핑된 메서드(showList)를 호출한다.

이때 DisasterService의 fetchVolcanoData()를 호출한다. fetchVolcanoData()는 volcano.csv 파일을 파싱한 Volcano 객체 리스트를 반환한다.

model.attribute(“data”, volcanoes); 란 volcanoes 리스트를 data라는 이름으로 뷰(HTML)에 넘겨준다는 의미이다.

model.attribute(“type”, volcano); 도 마찬가지이다.

lab2.html을 보면 type을 이용하여 이미지를 변경하고

data를 이용하여 반복문을 사용하고 테이블을 만든다.

이제 detailView로 넘어가보자,

localhost:8080/list/volcano에서 Detail을 클릭하면

lab2.html의 코드

<a th:href="@{/detail(

type=${type},

url=${item.url},

name=${item.name},

date=${item.date},

alertLevel=${item.alertLevel},

colorCode=${item.colorCode}

)}">Detail</a> 에 의해 DisasterController의 /detail 경로로 요청을 보낸다.

이런 파라미터 정보들을 받아서 DisasterController에서는 showDetail이라는 메서드를 실행한다.

화산과 지진의 파라미터가 다르므로 @RequestParam(required=false)를 이용하여 에러가 나지 않도록 하였다.

다시 model로 템플릿에 데이터를 전달하고 detailView.html 렌더링한다.

volcano의 url은 DisasterService에서 csv파일을 파싱하여 만들어졌었음.

url을 클릭하면 새로운 창으로 USGS에서 volcano 상세 정보가 출력 됨.

earthquake도 이와 동일한 방식으로 실행된다.